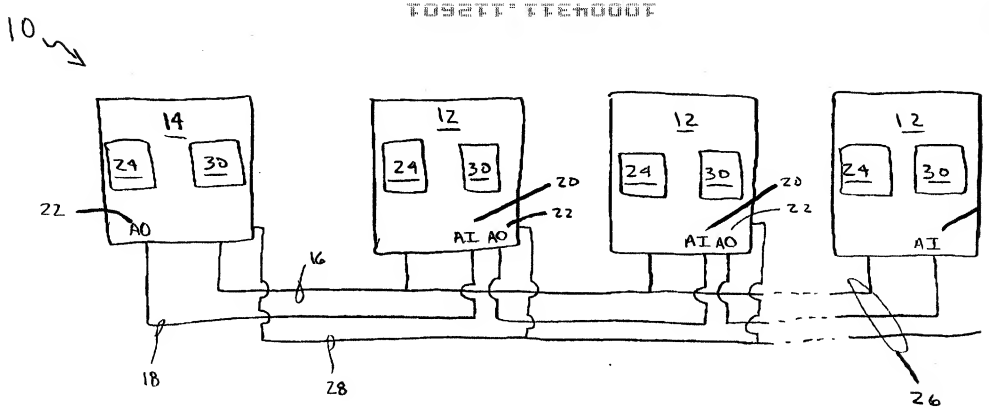
10
14
12
12
12
24
30
24
30
24
30
24
30
20
22
20
22
22
AO
AI AO
AI AO
AI
16
18
28
26

FIGURE 1

1. $TEMP_toggle is set equal to $MAXNODES
2. $TEMP_toggle transitions, from $INACTIVE to $ACTIVE, are exhibited on its address output pin. The pin is left $ACTIVE.
3. The message $GETCOUNT($DEFAULT) message is sent.
4. If the message succeeded in reaching the node:
    - 4.1. The value found in the reply to the message is stored in $TEMP_count. {Since the output pin of the server node is left $ACTIVE, the client node immediately adjacent to it has $NODEID equal to $UNCONFIGURED, the adjacent node responds as node $DEFAULT}
    - 4.2. $TEMP_node is set to ($TEMP_count +1)
    - 4.3. The message $SETADDRESS($DEFAULT, $TEMP_node +1) is sent.
    - 4.4. The server node's output address pin is set $INACTIVE.

{At this point, the node immediately adjacent to the server node has been given its address}

- 4.5. $TEMP_toggle is set to ($MAXNODES-$TEMP_node)
- 4.6. The server sends the message $TOGGLE($TEMP_node, $TEMP_toggle)
- 4.7. The server sends the message $GETCOUNT($DEFAULT).
- 4.8. If the message succeeded in reaching the node:
    - 4.8.1. The resulting value is stored in $TEMP_count. {Since the output pin of the node $TEMP_node is $ACTIVE, the client node immediately adjacent to it has $NODEID equal to $UNCONFIGURED, the adjacent client node responds as node $DEFAULT}
    - 4.8.2. $TEMP_newnode is set to ($TEMP_node + $TEMP_toggle - $TEMP_count +1).
    - 4.8.3. The server sends the message $SETADDRESS($DEFAULT,$TEMP_newnode)
    - 4.8.4. The server sends the message $SETADDRESSPIN($TEMP_node, $INACTIVE)
    - 4.8.5. $ TEMP_node is set equal to $TEMP_newnode
    - 4.8.6. Go to step 4.5
- 5. Finished; all adjacent nodes have had their addresses assigned

FIGURE 2

1. $TEMP_node is set to 0.
2. If $TEMP_node is equal to $MAXNODES, then exit {no unconfigured node found}.
3. The server sends the message $GETCOUNT($TEMP_node+1)
4. If the message succeeded in reaching the node:
    4.1. $TEMP_node =$TEMP_node +1
    4.2. Return to step 2

(At this point, the $TEMP_node is the address of an existing node with no immediate successor)

5. $TEMP_toggle is set to ($MAXNODES-$TEMP_node)
6. If $TEMP_node is equal to zero
    6.1. The server toggles causes $TEMP_toggle transitions from $INACTIVE to $ACTIVE to be exhibited on its output address pin, and the pin is left $ACTIVE
    6.2. Else {the case in which $TEMP_node not equal to zero}
        6.2.1. The server sends the message $TOGGLE($TEMP_node, $TEMP_toggle)
7. The server sends the message $GETCOUNT($DEFAULT).
8. If the message succeeded in reaching the node:
    8.1. The resulting value is stored in $TEMP_count. {Since the output pin of the node $TEMP_node is $ACTIVE, the client node immediately adjacent to it responds as node $DEFAULT}
    8.2. The server sends the message $SETADDRESS$DEFAULT, $TEMP_node + $TEMP_toggle - $TEMP_count +1)
    8.3. If $TEMP_node is equal to 0
        8.3.1. The server sets its output address pin to $INACTIVE
        8.3.2. Else {the case in which $TEMP_node is not equal to zero}
            8.3.2.1. The server sends the message $SETADDRESSPIN($TEMP_node, $INACTIVE)

Else (the message did not reach the node, meaning the node that made the broadcast is not adjacent to the node whose $NODEID is $TEMP_node)

9.1. Set $TEMP_node equal to $TEMP_node +1
9.2. If $TEMP_ node is greater than $MAXNODES, then exit {no unconfigured note found}
9.3. The server sends the message $GETCOUNT($TEMP_node)
9.4. If the message succeeded in reaching the node:
    9.4.1. Return to step 2
    9.4.2. Else {no node found}
        9.4.2.1. Got to step 9.1

10. Finished

FIGURE 3

Upon receipt of the address request from the client node 12, the server 14 performs the following steps:

1. $TEMP_node is set to $OLDEST.
2 If $TEMP_node is equal to $UNCONFIGURED, then exit
{no unconfigured node found}.
3. The server sends the message $GETCOUNT($TEMP_node)
4. If the message succeeded in reaching the node:
4.1 $TEMP_node = $ACT[$TEMP_node].$NEWER
4.2 Return to step 2

{At this point, the $TEMP_node is the address of a node which is not responding and which has not been heard from for the longest time}

5. If $TEMP_node is equal to $UNCONFIGURED, then exit {no unconfigured node found}.
6. $TEMP_node = $TEMP_node - 1
7. If $TEMP_node is greater than 0
7.1 The server sends the message $GETCOUNT($TEMP_node)
7.2 If the message succeeded in reaching the node:
7.3 Return to step 6

{At this point, the $TEMP_node is the address of an existing node with no immediate successor}

8. $TEMP_toggle is set to ($MAXNODES - $TEMP_node)
9. If $TEMP_node is equal to zero
9.1 The server toggles causes $TEMP_toggle transitions from $INACTIVE to $ACTIVE to be exhibited on its output address pin, and the pin is left $ACTIVE
9.2 Else {the case in which $TEMP_node not equal to zero}
9.2.1. The server sends the message $TOGGLE($TEMP_node, $TEMP_toggle)
10. The server sends the message $GETCOUNT($DEFAULT)
11. If the message succeeded in reaching the node:
11.1 The resulting value is stored in $TEMP_count. {Since the output pin of the node $TEMP_node is $ACTIVE, the client node immediately adjacent to it responds as node $DEFAULT}
11.2 The server sends the message
$SETADDRESS($DEFAULT, $TEMP_node+$TEMP_toggle-$TEMP_count+1)
11.3 If $TEMP_node is equal to 0
11.3.1. The server sets its output address pin to $INACTIVE
11.3.2 Else {the case in which $TEMP_node is not equal to zero}
11.3.2.1 The server sends the message $SETADDRESSPIN($TEMP_node, $INACTIVE)
12. Else {the message did not reach the node, meaning the node that made the broadcast is not adjacent to the node whose $NODEID is $TEMP_node}
12.1. Set $TEMP_node equal to $TEMP_node+1
12.2. If $TEMP_node is greater than $MAXNODES, then exit {no unconfigured node found}.
12.3. The server sends the message $GETCOUNT($TEMP_node)
12.4. If the message succeeded in reaching the node:
12.4.1. Return to step 2
12.4.2. Else {no node found}
12.4.2.1. Got to step 9.1

13. Finished.

FIGURE 4

Updating the activity table, $ACT.

1. Store the node id of the sender/receiver in $TEMP_node
2. $TEMP_older=$ACT[$TEMP_node].$OLDER
3. $TEMP_newer=$ACT[$TEMP_node].$NEWER
4. If $TEMP_older is equal to $UNCONFIGURED
    4.1. Then set $OLDEST equal to $TEMP_newer
    4.2 Else $ACT[$TEMP_older].$NEWER=$TEMP_newer
1. If $TEMP_newer is not equal to $UNCONFIGURED
    5.1 Then set $ACT[$TEMP_newer].$OLDER=$TEMP_older
1. $ACT[$TEMP_node].$NEWER=$UNCONFIGURED.
2. $ACT[$TEMP_node].$OLDER = $NEWEST
3. $NEWEST = $TEMP_node

FIGURE 5

Mapping of CANOpen constructs for Automatic Addressing

Temporary variables	Each note has storage available for the temporary variables required to support the addressing scheme.
$TIME	Each node has a timer, implemented as a memory location which is incremented each time a periodic interrupt occurs. The granularity of this timer is not critical to the addressing scheme, but the time-related tuning parameters will be some multiple of the basic timer increment.
$DEFAULT	The default address to be taken by a node shall be 127.
$UNCONFIGURED	The value stored in a node which has no address configured shall be 0.
$ACTIVE	The value corresponding to the active state of an addressing pin shall be any voltage less than -4.5 volts.
$INACTIVE	The value corresponding to the inactive state of an addressing pin shall be any voltage greater than -0.5 volts.
$ACTIVETIME	The duration of the active portion of a cycle on the address line shall be two clock ticks.
$INACTIVETIME	The duration of the inactive portion of a cycle on the address line shall be two clock ticks.
$TIMEOUT	The timeout period after a series of state changes on an address pin shall be 4 clock ticks.
$ADDRESSMSGTIMEMIN	The server must complete a successful addressing sequence within 32 + 2*N clock ticks of the $TOGGLE message completion, where N is the number of transitions specific in the $TOGGLE message.
$ADDRESSMSGTIMEMAX	The server must not begin another addressing sequence until 64 + 2*N clock ticks have passed since a previous $TOGGLE message was sent.
$SETADDRESS(X,N)	This corresponds to CANOpen LSS service
$SETADDRESSPIN(X, Y)	This is implemented by writing to a node OD (using a CANOpen SDO) at a determined index, subindex 1. When this OD entry is written with 0, the address pin is set to $INACTIVE, otherwise it is set $ACTIVE.
$TOGGLE(X, N)	This is implemented by writing to a node (using a CANOpen SDO) at a predetermined index, subindex 2. When this entry is written, the node should begin producing transitions on its output address pin.
$GETCOUNT(X)	This corresponds to reading from a node (using a CANOPen SDO) at a determined index, subindex 3. When a node detects an $ACTIVE-going transition on its input address pin, it should record it in this entry.
$NODEID	This value is stored in each node.
$COUNT	This is a storage location in each node, at determined, subindex 3.
$TIMESTAMP	This is the current value of the clock in each node.
$ADDRESSREQUEST	This is implemented using CANOpen LSS layer service.

FIGURE 6